DIGITAL TRANSFORMATION IN HIGHER EDUCATION: SELECTION, TEST AND ACQUISITION OF A BUSINESS SUPPORT SYSTEM – EXPERIENCES FROM THE FIELD AND LESSONS LEARNED

T. Richter, S. Baum, S. Böhmer, S. Klemenjak, A. Roettgen, C. Stich, M. Vahl, F. Westerfeld

University of Applied Sciences Bonn-Rhein-Sieg (GERMANY)

Abstract

Digital transformation in Higher Education and Science is a mission-critical demand to prepare educational institutions for their future competition on the international market. In many cases, the digitization goes along with the search for and acquisition of new software. For easily exchangeable software, wrong product decisions, in the worst case, lead to calculable financial losses. However, if a planned software requires a lot of technological adjustments and is to be applied as central component of a business- and/or security-critical environment, wrong decisions during the software acquisition process might lead to hardly calculable damage. Questions arising are how to decide for a product and how many resources should be invested for the acquisition process.

We planned to apply a commercial Business Support System, which should replace the currently used in-house developed software. Our goals were the increase of our university's level of data security, to ease the interaction between stakeholders, to eliminate media discontinuities, to improve the process management and transparency, and to reduce the execution time of automated processes. Alongside with the introduction of the electronic case file, our agenda stipulates the digitization (and automation) of administrative university processes, especially, but not limited to, the student self-service and the administrative student life cycle. Usual tools and practices, commonly applied to (simple) software acquisition, failed in our scenario.

With the case study introduced in this paper, we address all persons, involved within software acquisition processes: From our experiences, we strongly recommend to place greater value on an exhaustively completed acquisition process, than on short-termed economic advantages.

Keywords: Business Support System, Process Automation, Student Life Cycle, Student Self-Service, Student Administration, Software Acquisition.

1 INTRODUCTION

In Europe, the digital transformation in Higher Education (HE) and Science is considered mission-critical for the future competition on the international market. Several related national and European funding programs and initiatives have been launched, in order to foster cooperation, internationalization, interoperability, transparency, accountability and openness within and across public administrations, enterprises, organizations and institutions (in the following: "organizations"). Measures to being implemented, range from standardization of IT-infrastructures, services and processes and related technological innovations, electronic file management, the establishment of digital communication, cooperation and data sharing platforms, and programs, fostering the digital literacy of European citizens [1]. The improvement of customer portals and subsequently, the digitization of business-processes within organizations is understood as a significant milestone within the proposed European digital strategy [2]. Even though digital transformation can include a variety of different measures, it is not an end in itself: A digital transformation should generally be understood as a development process, beginning at an organization's individual starting point, progressing according to the organizations' digital agenda and maturity, and ending with the intended level of digitization.

Even though heterogeneous IT-infrastructures within an organization require higher efforts for maintenance, they are not problematic in themselves, as long as it is possible to access data and carry out control across the various systems. Even if a strict isolation of processes, systems and data is intended due reasons of security management, at least some of the data within the system actually remain indispensable for other units in order to successfully complete their work [3]. Without mechanisms to maintain the necessary data access across systems, media discontinuities arise as a consequence, which lead to time-consuming and potentially fault-prone manual work, up to paper-

based intermediate steps, necessary to transfer data from one to another IT-system. Interoperability between productive IT-systems gets particularly eminent when organization-wide processes are to be digitized. In our own case, amongst others, we planned to digitize parts of the administrative and supportive processes for our students (as a part of a students' portal). Such processes require data from and need to involve people within distinct organization units, such as faculties, student registrar's office, examination authority, financial office and international office. While in our university, many different systems (hardware and software) run simultaneously, data exchangeability, institution-wide messaging and data exchange constitute basic requirements to reach our goal.

There are strategies to solve the general problem of lacking interoperability between IT-systems, just to name the most intrusive one, a consequent technological harmonization of the whole IT-infrastructure, Software (SW) and Hardware (HW). Such a roll-out within a productive environment generally is very complicated to organize (without disturbing the daily business), cost-intensive, and bears a high risk of technical and psychological failure. Amongst others, especially the lack of user acceptance, when beloved applications shall be substituted, must not be underestimated [4].

A less intrusive approach is the integration of a middleware solution which often is a sufficient alternative, when the focus lies on interoperability of IT-systems and exchangeability of data. Middleware, as an additional layer in the IT-infrastructure, is a special type of software, designed to foster the interaction across different autonomous systems. Strongly simplified, standard interfaces are used to submit messages and data to and between the involved IT-systems. In this sense, middleware provides services to harmonize the various involved systems on a pure logical basis, fully keeping the user ignorant of the actual technological finesse behind it. Most Business Support Systems (BSSs), on which we focus in our case study, applied to digitize business processes, either make use of an existing middleware or provide own middleware functionalities.

Today's advertisement and commercial "semi research" promote acquisition of business-SW, such as Enterprise Resource Planning (ERP) Systems or BSSs as a rather simple decision process: The Gardner Magic Quadrant (GMQ) [5] appears to be the most popular analytical tool in this context. It positions products of a selected branch within a construct of four quadrants, on the X-axis regarding their completeness of vision and on the Y-axis, on a likewise undefined scale, regarding their "Ability to Execute". Unless its prominence in the field, the construction of and the theory behind the GMQ lack transparency, particularly regarding the questions, why one product has been considered for positioning and others not, if products were at all excluded due to not having met certain criteria, which criteria actually led to positioning, how the positioning has taken place (a question of the scale) and how/if the criteria are modified between the yearly reports. Snapp [6] claims that the main criteria are the size of a company and its willingness to pay for the evaluation. As for the 2018 GMQ for "Intelligent Business Process Management Suites" [7], mainly big vendors actually can be found, which appears to confirm his claim. However, even though the producers' persistency on the market and the level of innovation of a product might be relevant criteria, for purchasing business-critical software, far more investigations and arguments are required. In this paper, we first subsume the state of the art regarding research in SW acquisition, then introduce our own case and afterwards reflect on experiences and provide recommendation for future SW acquisition processes.

1.1 Terminology

In our research for "ready-to-use" BSS-solutions, we found some terms in the literature not consistently defined. We define the terms "Workflow", "Business Process", "Business Process Management System", "Workflow Management System", and "Business Support System", as follows.

1.1.1 Workflow

A workflow defines within a process, who needs to do what within which time, who needs being informed, who confirms (conditions) activities, and which information need to be archived.

1.1.2 Business Process

A business process consists of logically combined activities and links to other business processes, necessary to reach a desired business objective [8]. For Business processes, two different levels can be distinguished, i.e., the process level with a logical/managerial perspective (Fig.1), and the activity level, which describes the activities that are to be carried out in a defined sequence.

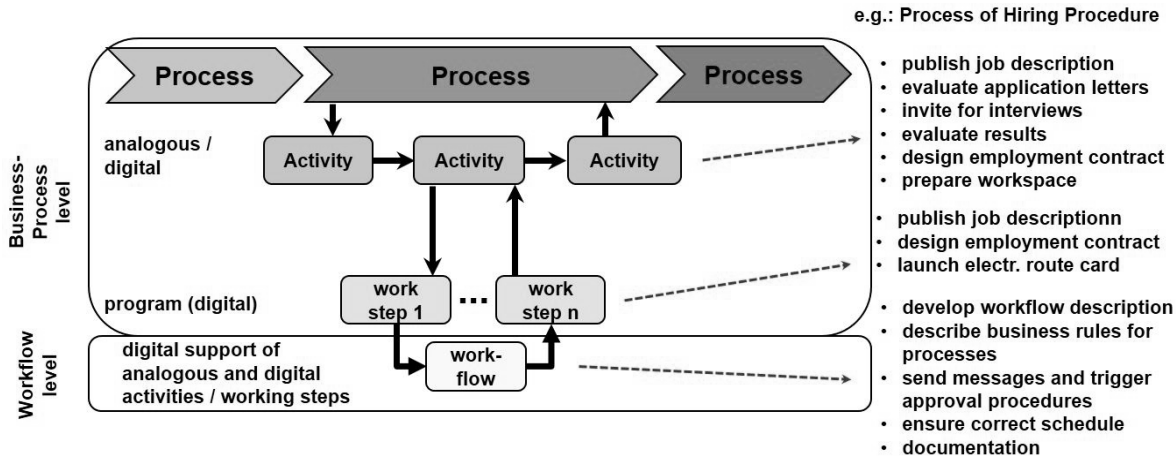

Figure 1. Workflows and Business Processes.

Figure 1 illustrates the difference between workflows and business processes alongside the example of a usual personnel hiring process. While a business process describes all sub-processes within a central business area and, on a rather abstract level, defines what has to be done in order to reach the aim, a workflow, precisely defines all operations necessary to move from one work step to the next.

1.1.3 Business Process Management System (BPMS)

A BPMS works on the "meta-level" of business processes and not on the operational level of workflows. In that sense, a BPMS mainly focuses on the analysis and control (management) of workflow-outcomes (status quo, successfully finalized, stopped because of error), on escalation protocols (in order to determine possible threats) and on automated messaging (in case of escalation). For this tasks, the full functionality of a workflow engine is not necessary, since receiving data from and about (partly manual) business processes, is sufficient. Also not necessary is a user interface for anyone else, but the managers who control the business processes.

1.1.4 Workflow Management System (WFMS)

Different to a BPMS, a WFMS cares for the operation of processes in detail. Thus, the heart of any WFMS is a workflow engine, providing the tools and mechanisms to model, implement, test and run workflows on operational level and manage visibility/accessibility rules according to (locally) defined user rights. Since workflow engines usually need to access data, produced from different systems, they need middleware functionalities in terms of explicitly implemented connectors and/or standardized interfaces (e.g., REST, SOAP). WFMSs usually also provide a development environment (a so called "designer studio") and necessarily, a user interface to initiate, maintain and stop available workflows. User-inputs usually are proceeded based on defined forms. Typical processes, supported/ automated through a WFMS, are a route card (sending process demands to the different responsible departments and collecting information about status quo) an order for procurement, an application of employees for leave, the inscription for a study program and demand for a status quo report on the current study progress of a student. Most WFMSs additionally have an own access management and provide light-weight functionalities regarding the analysis of workflows, which (per workflow-step) make information available to developers, regarding critical aspects and bottlenecks and to users, regarding the status quo (who has when done what) of a currently running workflow. Finally, WFMS provide tools for documentation, on the one hand, regarding process implementations (model, code, rulesets), on the other hand, process results (who, what, when, outcome).

1.1.5 Business Support System (BSS)

A BSS, first of all, unites a BPMS with a WFMS und thus, covers the functionalities from both perspectives, operational and managerial. Usual BSSs on the market have an own database, in which processes, documentation and reports, but also documents (as attachments of processes) are stored. Most BSSs also provide a user account management, which can be used as or be connected to a system-wide authentication provider, often including Single Sign On functionality (SSO). Furthermore, BSSs mostly offer basic functionalities of Document Management Systems (DMS), up to the opportunity, to support the full legal requirements for computerized accounting and tools to control the

storage of and handling with personalized data, according to legal demands (such as the General Data Protection Regulation, GDPR, on European level).

1.2 The Software Acquisition Process in Theory

Since 30 years, theory and practice of SW acquisition processes are constantly investigated issues: Past publications introduce generic approaches and models for acquisition processes [9] [10], analyse influences on related decision processes [11] or provide a meta-analysis on past publications in the field [12]. Others suggest concrete decision criteria [13] and critical factors during the decision process [14]. Some focus on the so-called "make, rent (in-house vs. outsourcing) or buy decision" [15] [16], analyse single-cases [17] or present comparative case studies, differentiate between specific types of SW according to the field of application (e.g. ERP) [18] [19] or concentrate on a SW's relevance for the organization (critical vs. non-critical) [20]. Further publications in the field focus on specific issues regarding open source projects [21].

For the decision process, Fowler [22] recommends an iteration of a four-step process for any stage of software design and development, taken during the development process, namely "*plan, execute, review and report*" (p. 52). Transferred to the SW acquisition process, the iteration would apply to the phases "requirements definition", "product research", "short list generation and testing", and "decision making and documentation". For the acquisition of exchangeable standard SW, the usual selection criteria cover required basic functionalities, price and the applied price model and cost of staff training. For SW, being used over a long time in a business-critical environment, additional aspects need to be incorporated, such as customizability (adaptability), system integration, stability of the software supplier on the market, conformity to local laws, regulations and policies, risks and risk management, and other factors. Originally applied to the context of hardware assurance, the Defect Detection and Prevention approach [23] is nowadays also used for preparing early life-cycle decisions in acquisition processes for business-critical SW [24]. Before taking such critical decisions, Feather et al. [9] strongly recommend not only to focus on critical requirements, but to additionally consider risks on general level and related mitigations (how to overcome the risks). Ponsard et al. [25] support this claim, emphasizing that "*verification and validation should be considered early in the development process*" (p. 233). They suggest the early definition of test cases regarding goal-oriented requirements and a comprehensive test phase, to ensure that these actually are met. Simmons [26] claims that the effectivity of Software Quality Assurance (SQA) massively increases, the earlier it is launched within the acquisition process. After an analysis of typical SW trouble spots during the SW acquisition process, Simmons concludes that for a successful SQA, it is crucial to involve the later users far beyond the requirements analysis. Anyways, involving the later users early in the process is a matter of user acceptance: Brannigan suggests to integrate the later users into the software acquisition test-cycles [27]. Technology acceptance is a subject for investigations since many years and nowadays, understood as a critical factor for a successful implementation of new technologies in enterprises. The Technology Acceptance Model (TAM), meanwhile a standard tool to determine issues of user acceptance [28], has continuously been extended over the years and, in its current third version, has been extended with a special focus on SW. This third version considers determinants, critical for the users' willingness to accept changes [29]. Jeffrey transferred the TAM-3 into the context of Higher Education and technology enhanced learning [30] and found that even though the considered influence factors to some extent, still play a role for user acceptance, their relevance fades "*with greater fluency, more extensive use of computers, and the effect of digital wisdom*". He came to the conclusion that fostering the user acceptance in the context of higher education, severely reducing the factors considered within the TAM-3 would at least lead to the same results.

1.3 The Scenario of our Case Study

A new version of our general student administration system was released and has to be applied. The SW included a massive database redesign, a new user interface for students and administration, and for our case most relevant, a set of formerly unavailable functionalities to automatize processes in the context of student administration and student self-service. So far, we already had digitized many internal processes through an in-house developed BSS, implemented 18 years ago and worth modernization from both, a technological and operational perspective. For further motivation, our whole network infrastructure has (physically) been redesigned and modernized. Additionally, the SW-architecture paradigm within the university's administration has been shifted. Currently, employees and students have to deal with a variety of decentralized accessible applications, services and data access points to manage their daily work. For the near future, we planned to create an application-

independent presentation layer, providing a personalized single access point for all individually required applications and services. Finally, triggered through the management, a fundamental paradigm change regarding the make-or-buy decision for in-house used SW had taken place. While before, in-house produced SW was welcome in our university, the management decided to rather rely on the benefits of standard SW, which amongst others, were reduced dependency from a small number of highly specialized developers, the responsibility of an external producer for regular bug-fixes and functional updates, a significant reduction of maintenance time, a reasonable community of users and thus, less individual incidents from SW-errors, the availability of external consulting and development support and the availability of professional trainings for users and developers.

With all these planned and already implemented changes in our technical and logical infrastructure, we recognized a great chance to also substitute our BSS through a standard SW in the same period.

In preparation of the rollout of the first modules of the new student administration software, we closely cooperated with all of the involved departments and stakeholder-groups, investigating and documenting their current processes and possible potential for improvement. During this process analysis phase, we took the chance and also involved our software developers to determine necessary functional requirements for an alternative commercial BSS. With the resulting requirements-list at hand, we started to look for potential candidates on the global market.

As a public institution within Europe, we are forced to open a competitive bidding as soon, as the expected expenses exceed certain limits (on national or even European level). Depending on the level of competitive bidding range, the realization time for the project would need to be calculated. Thus, first of all, we had to know the market for BSS and particularly, the expectable price-range (and opportunities) for related products. We started with a very broad, international market analysis on what software generally was available in the field and which one could provide the best possible support.

2 THE SOFTWARE SELECTION PROCESS

The SW selection process included a requirement analysis, a market study, and the product selection.

2.1 Requirements Analysis

We started our acquisition process with systematic requirements analysis (see Fig. 2, 1.a). In this phase, we invited members of all groups of stakeholders who, in the future, would have to deal with the BSS, to contribute to the collection of requirements in a brainstorming-like process. Afterwards we aggregated the collected suggestions (eliminating doublets and subsuming similar requirements) and reduced the number of items through exclusion of extremely specific requirements (if non-essential). Eventually, we clustered the items into (thematically ordered) requirement-classes (e.g., financial issues, technological requirements, legal issues). For the further steps, we were able to reduce the involvement of users to the groups of managers, developers and administrators. We defined the full customisability of the user interface as an essential requirement, regarding both its functionalities and design. In such a setting, the user interface, which usually would have to be evaluated by the end-users according to its usability, can be fully adopted to the end-users' needs. Thus, even though meeting particular task-related requirements of the end-users is crucial to trigger the necessary user-acceptance for the productive system, it was not directly relevant for our product choice. We subsequently asked the management, developers and administrators to evaluate the items in the list of remaining requirements regarding their relevance (Fig. 2, 1.b), using the key-words "*essential*" (obligatory demanded), "*high*" (quite relevant but dispensable), "*medium*" (comfortable but not crucial) and "low" (nice to have). Further requirements, defined as essential were the availability of the SW in a German or English language version (on all layers, development, operational business and management), the opportunity for on-premises installation and data storage, the full support of our data privacy laws, and from technological perspective, openness regarding a heterogeneous IT-infrastructure, and the opportunity to add own functionalities and configure existing ones without interfering with the general workflow engine or losing update capability. Requirements, with a relevance defined as "high" were, amongst others, a price model, favouring one-time-costs (capital expenditure, capex) over repetitive costs (operational expenditure, opex), stability of the product/producer on the market (either the company is big enough to ensure the further existence for the next decade with a high probability, or the community is big enough and able to buy the whole product continuing its maintenance), and the support of a centralized access management. Requirements assigned to a medium and low relevance had less or even no impact in the initial product evaluation phases (Fig. 2, 1.d; 2.e; 3.d).

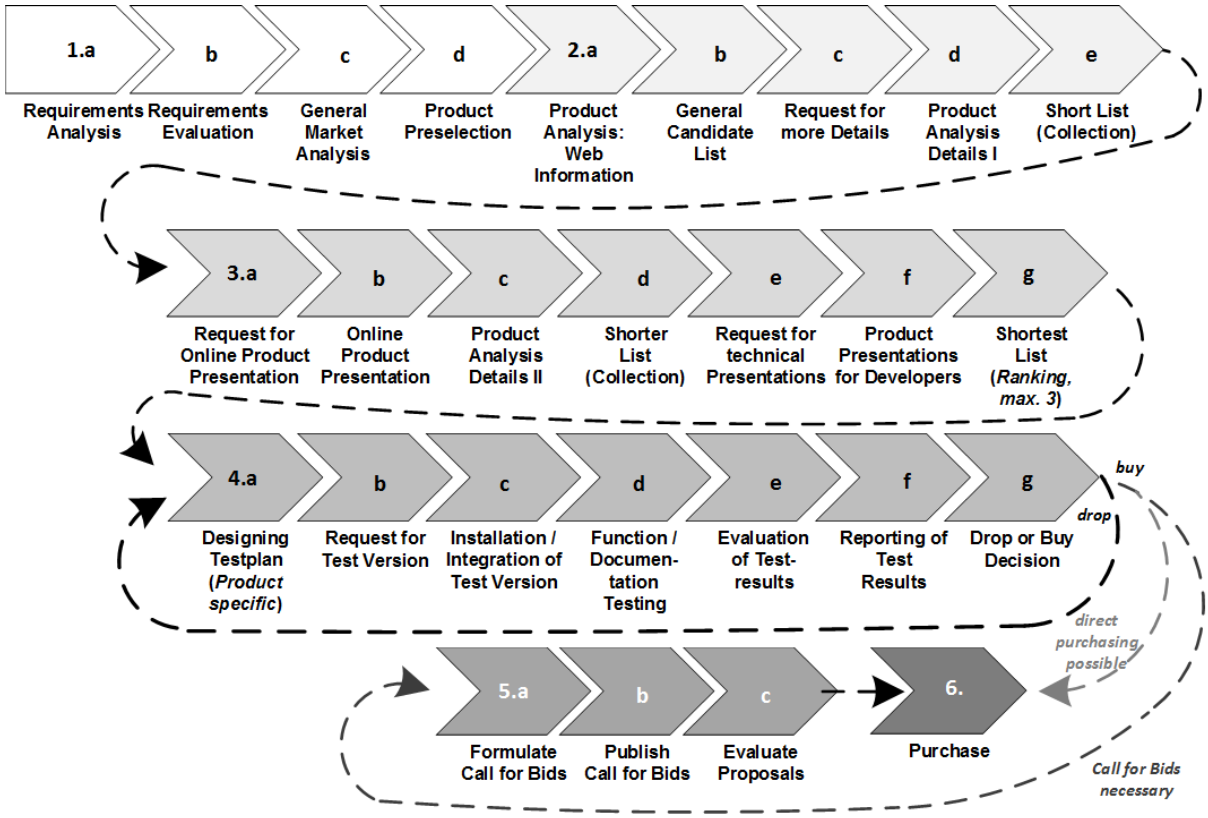

Figure 2. Product Acquisition Process.

2.2 Market Study – BSS Products on the International Market

In our investigation of the international market for BSSs Fig.2, 1.c), we found more than 70 products and we are sure that this list was not complete. We were aware that our comprehensive requirements list would not fully be met by any of the products. Some of the found products, just provided a couple of workflow management functionalities, implemented as useful extension to a company's main product (e.g. a Data Warehouse) or revealed as pure WFMSs without further BSS functionalities. In a first product preselection round (Fig. 2, 1.d), we removed these products from our list of potential candidates without further checks. For the rest, we collected all online available product information and customer-evaluations (Fig. 2, 2.a) and contrasted them with our requirements. As a result of this first analysis for regarding our essential requirements, products were excluded from the list, when they extremely exceeded our set price limit, exclusively were available as externally hosted cloud versions, exclusively were offered as SaaS (Software as a Service) on the basis of annual license fees (opex), and/or had cross-dependencies to third-party products, we did not use. The 23 remaining products were included in a General Candidate List (Fig. 2, 2.b). The information, most producers provided on their websites, was insufficient for a comparative SW analysis on the basis of our requirement list. Relevant details were missing, such as used technologies, applied implementation models and standards, development strategies, grade of openness regarding interfaces and own code, and price models. This constituted a central problem we had encountered during our online research.

2.3 Selection Process

We chose to conduct our more detailed analysis in several rounds. For each round, we defined exclusion criteria with a different focus.

2.3.1 Defining a Short List

We addressed the suppliers via e-mail with the request for more details on their products (Fig. 2, 2.c). Together with our questions, mainly on features and limitations, we additionally explained the infrastructure, in which the product would need being integrated. Quite a number of producers did not react at all or called us via phone. Most of these phone calls just were sales talks and answered none of our questions. Since we had to expect being similarly treated, once a contract is signed, we

renounced further attempts to get more information from these firms but instead, excluded their products from our list. The responses, received from the remaining candidates, were analyzed (Fig. 2, 2.d). Some stated that for our specific case, their product would not be an appropriate choice, because of technological incompatibilities and/or missing features. The candidates who finally met all essential criteria and most of the requirements with high relevance, were included in our short list (Fig. 2, 2.e).

2.3.2 Defining a Shorter List

For the next evaluation round, the remaining firms were invited to present their products in a webinar (Fig. 2, 3.a). We limited the webinar-duration to a maximum of one hour and defined a fixed period in which they should take place (one month ahead). Again, we sent a list of questions, we would like getting answered and announced an auditory of around ten people consisting of managers, developers, administrators and end-users. We finally asked the firms to present their products on the basis of a semi-formal modeled, non-trivial business process we are currently using. Some of the firms backed out after closer consideration. Eight candidates eventually agreed to hold the webinars. From the presentations (Fig. 2, 3.b), most held by sales persons, not a single one referred to the process provided by us, but instead, introduced a strictly linear and extremely simple designed process. All presenters concordantly claimed that our process example was too challenging for the short time, which we had to accept. The presentations suggested that modeling and implementing even our more complex processes would be possible without programming. Even though these examples were suitable to provide an impression about the look and feel of the SW, they were neither adequate to show how the BSS supports the development of rather complex processes, nor to show the limits and limitations of the BSS. In the hindsight, some of the presenters at least gave us an impression about the power, challenges and limitations of their BSS. For the analysis of the presentations, we discussed our impressions within the group of participants and documented the outcomes (Fig. 2 3.c). After a comparative evaluation of the results, four candidates clearly stood out from the field, defining our shorter list (Fig. 2, 3.d). At this point, the last Open Source projects were excluded because of our lacking faith into community work: For a business-critical long-term investment, we feared that Open Source projects might too easily be abandoned.

2.3.3 Defining a Shortest List

The online presentations were meant to address all groups of stakeholders. We still lacked knowledge about technological and financial specifics of the BSSs. So we invited the last four product suppliers to personally present their products in our university (Fig. 2, 3.e) within a time limit of two hours. As auditory we announced SW developers. Again, we sent our own exemplary process, technology-related questions, and a more detailed list of our technological requirements. Three of the four providers agreed to hold the presentation, one dropped out, considering their product not meeting the requirements to be integrated into our IT-infrastructure. After the presentations (Fig. 2, 3.f), we asked the eight participants to evaluate the products according to their personal impressions on the basis of a prepared questionnaire, and afterwards, we collectively conducted a moderated SWAT Analysis for each product (Fig. 2, 3.g): While two products promised applicability, there was a clear preference for one of them, basing on the provided “nice-to-have” features.

3 SOFTWARE TESTS

We now had a good impression about the products’ strengths and weaknesses. Anyways, we decided to additionally execute a number of SW-tests in a testing environment, particularly to reach a better understanding regarding usability and limitations. For that purpose, we designed a product-specific test plan (Fig. 2, 4.a), considering our context and the product specification/documentation and asked the supplier to provide us a fully functional test version of the software (Fig 2., 4.b). Two full-time developers were assigned to do the testing, coordinated by the project leader. The work had to be done besides the day-to-day business, so that we considered one day per developer and week as realistic and scheduled the test phase over a period of three months. In order to reduce the training period, we booked a three day workshop, provided from the supplier. A team of seven persons took part in the workshop (developers and project management).

Because of our limited resources, our test plan focused on the implementation of the really challenging process mechanisms and interfaces. We assumed that simple mechanics did not need being tested with a sophisticated product, and we already had seen some of the basic functionalities during the previous workshop. One of our developers was responsible for installation and integration of the SW (Fig 2., 4.c) and afterwards, assigned to conduct tests with self-written interfaces, in order to ensure

that the data, stored within our heterogeneous infrastructure, could faultlessly be addressed (Fig. 2, 4.d). The other assigned developer had the task to implement one complex process, which covered all processes and sub-processes around scholarships, from application to approval. As a special task, repeatedly required mechanisms were to be designed as reusable modules for a planned developer-toolbox. Our experiences regarding installation, usability of the developing environment, SW documentation, and applicability of own code was documented in detail (Fig. 2, 4.f).

3.1 Test Results

During the test-phase, we found (evaluation: Fig. 2, 4.e) that a number of required mechanisms were not natively included within or supported through the WF-engine. An example for such a missing "mechanism" was the field-type "from-to", particularly relevant for any processes with deadlines. While such a field intuitively "just" represents a period of time, it can have very different functionalities which need being addressed, in order to ensure a correct input-format for each of the specific scenarios. After some communication with the supplier and the producer, we found out that the BSS's concept did not support self-defined field types. Additionally, we became aware of a technical limitation regarding the number of form-fields embeddable within a process, which massively complicated the implementation of almost all of our larger processes. Last, we found significant gaps in the developer-documentation, so that integrating own code often just was possible through try and error. On this higher development level, neither the supplier, nor the producer made efforts to further support us. When we asked the supplier for an improvement of the developer-documentation, the producer uploaded some additional pages, but not enough to solve at least some of our problems. The supplier finally explained that the standard user, who actually is focused by the producer, rarely leaves the graphical development area and thus, would not require such a detailed developer-documentation.

At this point, for us, it became clear that we would not be able to use the BSS in the planned extent (Fig. 2, 4.g) and time. Consequently, we dropped this BSS as a possible candidate for us.

3.2 Further Steps

When we analyzed the second product on our shortest list, which would have been the next one to go through our practice tests, we realized that the workflow engine of this BSS fully relied on Open Source projects. Even though the BSS provided features, additional to those the Open Source packages did include, none of these were essential for us, but rather understood as "goodies". Thus, we decided to give the underlying Open Source project a chance for the next tests. For this decision, we revoked earlier resentments against Open Source projects in a business-critical environment insofar, as this particular SW already was persistent on the market since over 20 years, consequently further developed with fixed upgrade terms, used by huge stock-exchange listed enterprises, had an impressive number of long-time active community members, and was supported by a reasonable number of third-party consultants providing workshops and technical support (development and troubleshooting). We adjusted our test plan to the characteristics of the new BSS (Fig. 2, 4.a), downloaded and installed the necessary software packages and restarted our tests. Right now, the tests still are going on.

The installation was both, more time-consuming, because we could not install the system with a single click, but instead, had to download, install and configure a lot of distinct modules. However, because of a well-designed documentation, the task itself was manageable and eventually proceeded without serious trouble. In our tests, we so far found the Open Source BSS professionally developed and well documented on all levels. For the test period, we assigned five months.

4 CONCLUSIONS

When we started with the acquisition process in our university, digitizing business process was no unknown terrain for us. On the basis of the experiences with our in-house produced BSS, we already had a detailed idea on what we were looking for; from a technical and functional perspective. However, we neither expected the requirements analysis, nor reaching a decision for a certain product being easy and quickly accomplished tasks.

It took a lot of efforts to find out the details, on which features actually are needed to best possible support our end-users in their current situation and serve the university in the long run. For this task it was necessary, not just to focus on big processes and most prominent users, but particularly take "niche users" and their often invisible, but not less essential processes into account. In the practice, far

too many SW projects generate massive secondary expenses through readjustments or even fail completely, just because specific requirements of niche-players (and their business-critical processes) remained unconsidered during the acquisition phase. Together with end-users, developers, technical staff and management, we developed a ranked list of requirements which accompanied our search for the best matching BSS product.

We had the choice between over 70 BSSs on the international market on the one hand and a list of essential requirements, on the other. We further had a long list of mechanisms, the BSS should support and another with nice-to-have features. Contrasting this high number of products was a challenge, particularly because we often were not supplied with the information, required for a detailed analysis. Further products, even though called BSS, revealed focussing on process management, instead of providing functionalities to digitize processes. In view of the high number of alternatives, we excluded such products. In three decision rounds, we comparatively analysed the rest of the products against our essential requirements list, the usability for end-users and technological/technical aspects. In the end, two products remained for the final choice with a clear preference. At this point, we could have taken the final decision, but still felt unsure. We decided to apply practice tests in an environment, similar to the later productive setting. We defined a test plan and started our tests. First, the tests led to satisfying results. When it came to more complex mechanisms, involving self-written code, the chosen BSS revealed missing flexibility, an incomplete developer-documentation, and from the supplier, a lack of readiness to provide development support. Eventually, we had to discard the chosen candidate in order to avoid later complications. We started the practice tests with the Open Source product, which built the code-basis for the number two product on our list. The tests are not yet completed but the results, so far, promising.

One could argue that acquiring software should not take up too many resources. With this economic approach, the practice tests, we implemented after apparently having found the best possible product, can be considered beyond appropriateness. However, if we had chosen the first ranked product without further practice tests, the result would have been fatal: We now would “own” a product, which does not provide the support necessary to run our business and thus, would not reach the needed level of acceptance amongst its users.

With this case study, we would like to encourage managers, decision makers and employees who are involved in acquisition processes, to take the time required to ensure a related long-term investment into a business critical SW is justifiable from economic, technical and social perspective.

REFERENCES

[1] European Commission, *Digital Transformation: Industry*, 2019. Retrieved from https://ec.europa.eu/growth/industry/policy/digital-transformation_en

[2] E. Brucker-Kley, T. Keller, & D. Kykalová, „Prozessmanagement als Gestaltungshebel der digitalen Transformation?“, in *Kundennutzen durch digitale Transformation* (E. Brucker-Kley, D. Kykalová & T. Keller, eds). Springer Gabler, Berlin, Heidelberg, pp. 3-17, 2018.

[3] J. Kratz, „Vereinheitlichung des IuK-Systems bei der T-Systems International GmbH“, *Controlling*, vol. 15, no. 11, pp. 641-645, 2003.

[4] P. Mertens, „Schwierigkeiten mit IT-Projekten der öffentlichen Verwaltung“. *Informatik Spektrum*, 32(1), pp. 42-49, 2009.

[5] C., Howson, J., Richardson, R. Sallam, & A. Kronz, *Magic Quadrant for Analytics and Business Intelligence* Platforms, Gartner Report 2019, Gartner Inc., 2019. Retrieved from https://www.gartner.com/doc/reprints?id=1-65P04FG&ct=190125&st=sb

[6] S. Snapp, “*Gartner and the Magic Quadrant: A Guide for Buyers, Vendors and Investors*”, Scm Focus, 2013.

[7] Gartner Inc, peerinsights: Intelligent Business Process Management Suites, 2018. Retrieved from https://www.gartner.com/reviews/market/intelligent-business-process-management-suites

[8] J. Becker & D. Kahn, „Der Prozess im Fokus“, in J. Becker, M. Kugeler, & M. Rosemann (eds.), *Prozessmanagement. Ein Leitfaden zur prozessorientierten Organisationsgestaltung*. 6th edition, Berlin et al.: Springer, pp. 3-16, 2008.

[9] M.S. Feather, S.L. Cornford, K.A. Hicks, J.D. Kiper, & T. Menzies, "A broad quantitative model for making early requirements decisions", *IEEE Software*, vol. 25, no. 2, pp. 49-56, 2008.

[10] B. Farbey & A. Finkelstein, "Software Acquisition: A business strategy analysis", in *Proceedings of the 5th IEEE International Symposium on Requirements Engineering*, pp. 76-83, 2001.

[11] R. Palanisamy, J. Verville, C. Bernadas, & N. Taskin, "An empirical study on the influences on the acquisition of enterprise software decisions: A practitioner's perspective, *Journal of Enterprise Information Management*, vol. 23, no. 5, pp. 610-639, 2010.

[12] R. Kohl & Associates, "*Comparing Software Acquisition Models Against Each Other: The 'Build' vs. 'Buy' vs. 'Rent' Trade Study*", White Paper of the Defense Technical Information Center, USA, 2012. Retrieved from https://apps.dtic.mil/dtic/tr/fulltext/u2/a563434.pdf

[13] J.M. Padillo & M. Diaby, "Multiple-criteria decision methodology for the make-or-buy problem", *International Journal of Production Research*, vol. 37, no. 14, pp. 3203-3229, 1999.

[14] F. Fui-Hoon Nah, J. Lee-Shang Lau, & J. Kuang, "Critical factors for successful implementation of enterprise systems", *BPM Journal*, vol. 7, no. 3, pp. 285-296, 2001.

[15] M. Rubel, "Make, Buy, or Lease: the Software Acquisition Dilemma", *Code Magazine – VFP Conversion Papers* (wo. date). Retrieved from https://www.codemag.com/Article/030014/Make-Buy-or-Lease-the-Software-Acquisition-Dilemma

[16] P. Nelson, A. Seidmann, & W. Richmond, „Software Acquisition: The Custom/Package and Insource/Outsource Dimensions" *Advances in Computers*, vol. 47, no. C, pp. 341-367, 1998.

[17] Defense Innovation Board, "*Software Acquisition and Practices (SWAP) Study"*, 2018. Retrieved from https://innovation.defense.gov/software/CollectionId/18734/

[18] J. Verville, R. Palanisamy, C. Bernadas, & A. Halingten, „ERP Acquisition Planning: A Critical Dimension for Making the Right Choice", *Long Range Planning*, vol. 40, no. 1, pp. 45-63, 2007.

[19] M. Haddara, "ERP Selection: The SMART Way", *Proc. Technology*, vol. 16, pp. 394-403, 2014.

[20] P.A. Lindsay, "Improved Acquisition Processes for Safety-Critical Systems in the Australian Department of Defence", in *Proceedings of the 6th Australian Workshop on SCS and Software*, Australian Computer Society Inc., Darlinghurst, Australia, pp. 31–38, 2001.

[21] J. Holck, M.K. Pedersen, & M.H. Larsens, "Open Source Software Acquisition: Beyond the Business Case", in *Proceedings of the ECIS 2005*, pp. 128-141, 2005.

[22] K. Fowler, "Mission-Critical and Safety-Critical Development", *IEEE Instrumentation & Measurement Magazine*, vol. 7, no. 4, pp. 52-59, 2004.

[23] S.L. Cornford, M.S. Feather, & K.A. Hicks, "DDP-a tool for life-cycle risk management", in *IEEE Aerospace Conference Proceedings*, 2001. Retrieved from https://ieeexplore.ieee.org/document/931736

[24] M.S. Feather, J.D. Kiper, & T. Menzies, "Visualization Support for Risk-Informed Decision Making when Planning and Managing Software Developments", 2005. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/citations;jsessionid=FBA29009272431EFE6BCCE068A8E51D5?doi=10.1.1.515.2128

[25] C. Ponsard, P. Massonet, J.F. Molderez, A. Rifaut, A. van Lamsweerde, & H. Tran Van, "Early verification and validation of mission critical systems", *Electronic Notes in Theoretical Computer Science*, vol. 133, pp. 237-254, 2005.

[26] R.A. Simmons, "Software Quality Assurance (SQA) early in the Acquisition Process", in *Proceedings of the IEEE Conference on Aerospace and Electronics*, pp. 664-669,1990.

[27] V. Brannigan, "Acceptance Testing-The Critical Problem in Software Acquisition", *IEEE Transactions on Biomedical Engineering*, vol. 32, no. 4, pp. 295-299, 1985.

[28] H. van der Heijden, "E-TAM: A revision of the Technology Acceptance Model to explain website revisits", *FEWEB Research Memoranda*, 2000. Retrieved from http://hdl.handle.net/1871/1583

[29] V. Venkatesh & H. Bala. "Technology Acceptance Model 3 and a Research Agenda on Interventions", *Decision Sciences*, vol. 39, no. 2, pp. 273-315, 2008.

[30] D.A. Jeffrey, "*Testing the TAM 3 with the Inclusion of Change Fatique and Overload, in the Context of Faculty from Seventh-day Adventist Universities: A Revised Model*", Andrews Univ., School of Education, 2015. Retrieved from https://digitalcommons.andrews.edu/dissertations/1581